High Throughput Radix-2^{(K)} Feedforward Fft Architectures

Mohana Shankar V, Mageshkumar P, Vinoth kumar S. V, S K Binu Siva Singh

Electronics and Communication Engineering, Jeppiaar Engineering College, Chennai, India

ABSTRACT

The appearance of radix-2^2 was a milestone in the design of pipelined FFT hardware architectures. Later, radix-2^2 was extended to radix-2^k. However, radix-2^4 was only proposed for Single-path Delay Feedback (SDF) architectures, but not for feedforward ones, also called Multi-path Delay Commutator (MDC). This paper presents the radix-2^k feedforward (MDC) FFT architectures. In feedforward architectures radix-2^k can be used for any number of parallel samples which is a power of two. Furthermore, both decimation in frequency (DIF) and decimation in time (DIT) decompositions can be used. In addition to this, the designs can achieve very high throughputs, which make them suitable for the most demanding applications. Indeed, the proposed radix-2^k feedforward architectures require fewer hardware resources than parallel feedback ones, also called Multi-path Delay Feedback (MDF), when several samples in parallel must be processed. As a result, the proposed radix-2^k feedforward architectures not only offer an attractive solution for current applications, but also open up a new research line on feedforward structures.

1. INTRODUCTION

We are living in an era of electronics and communication. Since the development of long distance communication techniques in the 19th century, evolution of communication has always been striving towards higher data rates and better quality of information exchange. This has necessitated the invention of new and newer communication technologies in a very rapid phase. As advancement, it can be seen that most wired communication systems are being replaced with wireless communication systems [1-6]. Though wired transmission can offer much higher data rates, wireless solutions are preferred due to its flexibility and when communicating devices are mobile. Wireless communication is a rapidly developing high speed and high fidelity technology, which allows short range and long range transmission, enabling multimedia communications between portable devices. Initially wireless communication was popular only for cellular communications. Later a wide range of new wireless services such as cordless telephones, Wireless Local Loop (WLL) were introduced. These services were using analog signals and later revived to use digital signals. Since 2003, several advancements led to a fresh interest in wireless communications. Other than cellular and voice transmission, many other wireless services such as broadband wireless access systems, including the Wireless Local Area Networks (WLANs), Wireless Personal Area Networks (WPANs) and Wireless Metropolitan Area Networks (WMANs) were introduced. This domain has turned out to be a subject of extensive research and hence many standardization activities are undertaken throughout the world. Wireless applications require complex digital signal processing algorithms, which demand strong design constraints in terms of clock frequency, hardware resources, power consumption, latency, throughput and complex computations. To meet all these requirements, high speed and complex computation hardware fabric such as Field Programmable Gate Arrays (FPGAs) or Application Specific Integrated Circuits (ASICs) are required. These devices can be clocked at very high frequencies and they can be used to calculate sophisticated signal processing algorithms[7-10]. The Fast Fourier Transform (FFT) is arguably the most important signal processing algorithm in wireless application. FFT is a mathematical algorithm to efficiently compute Discrete Fourier Transform (DFT). FFT is an integral part of the Orthogonal Frequency Division Multiplexing (OFDM) based wireless systems. In an OFDM system, a very high rate data stream is divided into multiple parallel low rate data streams. Each smaller data stream is then mapped to individual data sub-carrier and modulated. The modulation of sub-carriers is performed by the FFT algorithm. The FFT size depends on the number of subcarriers used in the wireless standard. Considering a specific wireless standard, the Wimax, FFT length varies from 256 to 2048. In literature many different hardware architectures have been proposed for different wireless standards and implemented in FPGAs since the advent of wireless communications. The architectures are usually designed for very concrete design specifications in terms of signal length, precision, performance etc. Further, they target different wireless standards. In this thesis work, a reconfigurable FFT hardware architecture is proposed that can be applied to different OFDM-based wire to calculate FFTs of different lengths [11-16]. The proposed architecture is designed to support various FFT lengths, ranging from 64-point to 2048-point, for different OFDM wireless standards using single hardware architecture. The proposed scheme uses clock gating techniques and powers down unused hardware to minimize
power consumption. It is illustrated that the proposed architecture achieves a more flexible and simpler design in comparison with the architectures in literature [17].

1.1. DISCRETE FOURIER TRANSFORM

Discrete Fourier Transform (DFT) is a well-used and very powerful procedure in the field of digital signal processing. It is the discrete version of the popular continuous Fourier Transform.

**Definition (Discrete Fourier Transform)** the N-point DFT for a sequence \( x[n] \) is defined as

\[
X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}
\]

Where

\[
W_N^{nk} = e^{-j2\pi nk/N}
\]

And \( k = 0, 1 \ldots N-1 \).

The DFT transforms the complex discrete time domain signal \( x[n] \) to a complex signal in the frequency domain, \( X[k] \). The frequency-domain samples, the outputs of the DFT, are called “output bins” or simply “bins”.

All physical quantities are real-valued and are in many cases sampled into real numbers, in contrary to complex numbers. Since real numbers are a subset of complex numbers, the DFT can be calculated for real valued numbers. The DFT has an important property when it is calculated on a real-valued set of inputs. When the input sequence, \( x[n] \) where \( n = 0, 1 \ldots N-1 \), is real valued, the output of the DFT is conjugate symmetric \( X[k] = X[N-k] \), \( k=1 \ldots N-1 \). Where superscript * denotes complex conjugation. The complex conjugation of \( x = x_r + jx_j \) is defined as \( x = x_r - jx_j \).

The symmetry property shown means that the real part of the output sequence, \( X[k] \) where \( k = 0, 1 \ldots N - 1 \) has even symmetry and the imaginary part of the output sequence has odd symmetry. The magnitude of the output sequence is then even symmetric, \( |X[k]| = X[N-k] \), \( k=1 \ldots N-1 \). Worth noticing is the fact that \( X[0] \) is independent and do not have a symmetric sibling. When \( N \) is even output bin \( N/2 \), \( X[N/2] \), is also independent.

Because of the symmetry, only \( N/2 + 1 \) output values are independent for even numbered \( N \). The rest are redundant and can be calculated from the set of independent values. This means that a 1024-point FFT only gives 513 useful output samples when the input is real-valued.

1.2. FAST FOURIER TRANSFORM

A Fast Fourier Transform is an algorithm that calculates the Discrete Fourier Transform or its inverse in a more efficient way compared to direct calculation from the DFT definition. With the introduction of FFT algorithms, the use of DFT has become practical for large problem sizes, where the calculation of DFT are too time consuming.

The Cooley-Tukey algorithm describes a way to decompose an \( N = N_1N_2 \) point DFT into two separate \( N_1 \) and \( N_2 \) point DFTs. The radix-2 FFT divides an \( N \)-point DFT into two \( N/2 \)-point interleaved FFTs, which in turn is divided further until a set of the most basic 2-point DFTs are left. This 2-point DFT is the atom part of the radix-2 FFT and is called a butterfly operation. Using this decomposition, the FFT is calculated in a series of \( n = \log_2 N \) stages.

A radix-2 butterfly operation takes two input samples and calculates two output samples. The flow graph is shown in Figure 1.

**Figure 1:** Flow graph of a radix-2 butterfly element.

Decimation In Frequency (DIF) and Decimation In Time (DIT) are the two most common ways to decompose the Cooley-Tukey FFT. DIT separates the input sequence, \( x[n] \), into odd and even samples. Below is the general FFT of a 16 point FFT radix-2 algorithm Decimation in Frequency (DIF).

1.3. FEEDFORWARD FFT ARCHITECTURES

The \( N \)-point DFT of an input sequence \( x[n] \) is defined as:

\[
X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad k = 0, 1, \ldots, N - 1
\]

where \( W_N^{nk} = e^{-j2\pi nk/N} \).

When \( N \) is a power of two, the FFT based on the Cooley-Tukey algorithm [25] is most commonly used in order to compute the DFT efficiently. The Cooley-Tukey algorithm reduces the number of operations from \( O(N^2) \) for the DFT to
\( O(N \log_2 N) \) for the FFT. In accordance with this, the FFT is calculated in a series of \( n = \log_2 N \) stages, where \( \rho \) is the base of the radix, \( r \).

At each stage of the graphs, \( s \in \{1 \ldots n\} \), butterflies and rotations have to be calculated. The lower edges of the butterflies are always multiplied by \(-1\). These -1 are not depicted in order to simplify the graphs. The numbers at the input represent the index of the input sequence, whereas those at the output are the frequencies, \( k \), of the output sequence \( X[k] \). Finally, each number, \( \phi \), in between the stages indicates a rotation by:

\[
W_k^\phi = e^{-j \frac{2\pi}{N} \phi}
\]

As a consequence, samples for which \( \phi = 0 \) do not need to be rotated. Likewise, if \( \phi \in \{0, N/4, N/2, 3N/4\} \), the samples must be rotated by \( 0^\circ, 270^\circ, 180^\circ, \) and \( 90^\circ \), which correspond to complex multiplications by \( 1, -j, j, \) and \(-j\), respectively. These rotations are considered trivial, because they can be performed by interchanging the real and imaginary components and/or changing the sign of the data.

Radix-\(2^2\) is based on radix-2 and the flow graph of a radix-\(2^2\) DIF FFT can be obtained from the graph of a radix-2 DIF one. This can be done by breaking down each angle, \( \phi \), at odd stages into a trivial rotation and a non-trivial one, \( \phi' \), where \( \phi' = \phi \mod N/4 \), and moving the latter to the following stage. This is possible thanks to the fact that in the radix-2 DIF FFT the rotation angles of the two inputs of every butterfly, \( \phi_A \) and \( \phi_B \), only differ by \( 0 \) or \( N/4 \). Thus, if \( \phi_A = \phi' \) and \( \phi_B = \phi' + N/4 \), the rotation \( \phi' \) is moved to the following stage in accordance with:

\[
A e^{-j \frac{2\pi}{N} \phi'} \pm B e^{-j \frac{2\pi}{N}(\phi' + N/4)} = [A \pm (-j)B] e^{-j \frac{2\pi}{N} \phi'}
\]

Where the first side of equation (3) represents the computations using radix-2 and the second one using radix-2\(^2\), \( A \) and \( B \) being the input data of the butterfly.

### 1.4. PROPERTIES OF THE RADIX-2\(^2\) FFT ALGORITHM

<table>
<thead>
<tr>
<th>Properties Radix-2(^2)</th>
<th>DIF</th>
<th>DIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Butterflies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trivial rotations (odd s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( b_{n-s} \cdot b_{n-s-1} = 1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-trivial rotations (even s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( b_{n-s+1} + b_{n-s} = 1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( b_{n-s-1} + b_{n-s-2} = 1 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These are the properties for radix-2\(^2\) fft algorithm. Only when these conditions satisfy butterflies, trivial and non trivial rotations will be computed respectively.

The radix-22 feedforward architectures. First, 16-point 4-parallel radix-2\(^2\) feedforward FFT architecture is explained in depth in order to clarify the approach and show how to analyze the architectures. Then, radix-2\(^2\) feedforward architectures for different number of parallel samples are presented.

Figure shows 16-point 4-parallel radix-22 feedforward FFT architecture. The architecture is made up of radix-2 butterflies (R2), non-trivial rotators (x), trivial rotators, which are diamond-shaped, and shuffling structures, which consist of buffers and multiplexers. The lengths of the buffers are indicated by a number.

The architecture processes four samples in parallel in a continuous flow. The order of the data at the different stages is shown at the bottom of the figure by their indices, together with the bits \( bi \) that correspond to these indices. In the horizontal, indexed samples arrive at the same terminal at different time instants, whereas samples in the vertical arrive at the same time at different terminals. Finally, samples flow from left to right. Thus, indexed samples (0, 8, 4, and 12) arrive in parallel at the...
inputs of the circuit at the first clock cycle, whereas indexed samples (12, 13, 14, and 15) arrive at consecutive clock cycles at the lower input terminal.

Taking the previous considerations into account, the architecture can be analyzed as follows. Firstly, it can be observed that butterflies always operate in pairs of samples whose indices differ in bit $b_{n-s}$, meeting the property in Table I.

For instance, the pairs of data that arrive at the upper butterfly of the first stage are: (0, 8), (1, 9), (2, 10) and (3, 11). The binary representation of these pairs of numbers only differs in $b_3$. As, $n = 4$ and $s = 1$ at the first stage, $b_{n-s} = b_{4-1} = b_3$, so the condition is fulfilled. This property can also be checked for the rest of the butterflies in a similar way.

Secondly, Table I shows that rotations at odd stages are trivial and only affect samples whose indices fulfill $b_{n-s} \times b_{n-s-1} = 1$. By particularizing this condition for the first stage, $b_3 \times b_2 = 1$ is obtained. In the architecture shown in Figure 3 the indices that fulfill this condition are those of the lower edge and, thus, a trivial rotator is included at that edge. On the other hand, the condition for non-trivial rotations at even stages is $b_{n-s} + b_{n-s} = 1$. For all indexed samples at the upper edge of the second stage, this edge does not need any rotator. Conversely, for the rest of edges $b_2 + b_2 = 1$, so they include non-trivial rotators. The rotation memories of the circuit store the coefficients $\phi$ of the flow graph. It can be seen that the coefficient associated to each index is the same as that in the flow graph of Fig. 2. For instance, at the flow graph the sample with index $I = 14$ has to be rotated by $\phi = 6$ at the second stage. In the architecture shown in Fig. 3 the sample with index $I = 14$ is the third one that arrives at the lower edge of the second stage. Thus, the third position of the rotation memory of the lower rotator stores the coefficient for the angle $\phi = 6$.

Thirdly, the buffers and multiplexers carry out data shuffling. These circuits have already been used in previous pipelined FFT architectures [4], [17]–[20], and Figure 4 shows how they work. For the first $L$ clock cycles the multiplexers are set to “0”, $L$ being the length of the buffers. Thus, the first $L$ samples from the upper path (set A) are stored in the output buffer and the first $L$ samples from the lower path (set C) are stored in the input buffer. Next, the multiplexer changes to “1”, so set C passes to the output buffer and set D is stored in the input buffer. At the same time, sets A and B are provided in parallel at the output. When the multiplexer commutes again to “0”, sets C and D are provided in parallel. As a result, sets B and C are interchanged.
The above fig shows the proposed radix-2^2 feedforward architectures for the computation of the 64-point DIF FFT. Fig. show the cases of 2 4-parallel, and. These circuits can be analyzed as has been done for the architecture in Fig. 3. For this purpose, the order of the samples at every stage has been added at the bottom of the architectures. As can be seen in Fig. 5, in the proposed architectures the number of butterflies depends on to the number of samples in parallel, for any -parallel -point FFT the number of butterflies is. Therefore, the number of complex adders is. Likewise, the number of rotators is. The only exception is for. In this case, the number of rotators is. The proposed architectures can process a continuous flow of data. The throughput in samples per clock cycle is equal to the number of samples in parallel, whereas the latency is proportional to the size of the FFT divided by the number of parallel samples, i.e.,. Thus, the most suitable architecture for a given application can be selected by considering the throughput and latency that the application demands. Indeed, the number of parallel samples can be increased arbitrarily, which assures that the most demanding requirements are met. Finally, the memory size does not increase with the number of parallel samples. For the architectures shown in Fig. 5, the shuffling structure at any stage requires buffers of length. According to this, the total sample memory of the architectures is (4). Therefore, a total sample memory of addresses is enough for the computation of an -point FFT independently of the degree of parallelism of the FFT. Indeed, the total memory of addresses that the proposed architectures require is the minimum amount of memory for an -parallel FFT. Sometimes input samples are provided to the FFT in natural order and output frequencies are also required in natural order [27], [28]. Under these circumstances, reordering circuits are required before and after the FFT to adapt the input and output orders [27], [28]. For the proposed radix- feedforward FFTs the memory requirements for natural I/O depend on the FFT size and on the number of parallel samples. For a -parallel -point FFT a total memory of size is enough to carry out the input reordering, whereas a total memory of size is enough for the output reordering [24].
\[
P \cdot \left( \log_2 \frac{N}{k} - 1 \right), \quad \text{if } P < 2^k
\]
\[
\frac{2^k - 1}{2^k} \cdot P \cdot \left( \log_2 \frac{N}{k} - 1 \right), \quad \text{if } P \geq 2^k
\]

And the total memory is \( N \cdot P \). Likewise, the throughput is always equal to the number of parallel samples, \( P \), and the latency is \( N/P \).

2. SIMULATION AND OUTPUT

2.1. GENERAL RADIX 2^2 16 POINT FFT

Device utilization summary:
Selected Device: 5vlx50tf665-2
Slice Logic Utilization:
Number of Slice Registers: 311 out of 28800 \( 1\% \)
Number of Slice LUTs: 502 out of 28800 \( 1\% \)
Number used as Logic: 502 out of 28800 \( 1\% \)

Timing Summary:
Speed Grade: -2
Minimum period: 5.215ns
Maximum Frequency: 191.758MHz
Minimum input arrival time before clock: 1.814ns
Maximum output required time after clock: 2.826ns

2.2. FEEDFORWARD 16 POINT RADIX-2^2 FFT

Device utilization summary:
Selected Device: 5vlx50tf665-2
Slice Logic Utilization:
Number of Slice Registers: 1113 out of 28800 \( 3\% \)
Number of Slice LUTs: 1632 out of 28800 \( 5\% \)
Number used as Logic: 1536 out of 28800 \( 5\% \)

Timing Summary:
Speed Grade: -2
Minimum period: 6.766ns
Maximum Frequency: 147.790MHz
Minimum input arrival time before clock: 3.407ns
Maximum output required time after clock: 3.259ns

2.3. FEEDFORWARD 64 POINT RADIX-2^2 FFT

Device utilization summary:
Selected Device: 5vlx50tf665-2
Slice Logic Utilization:
Number of Slice Registers: 1113 out of 28800 \( 3\% \)
Number of Slice LUTs: 1632 out of 28800 \( 5\% \)
Number used as Logic: 1536 out of 28800 \( 5\% \)

Timing Summary:
Speed Grade: -2
Minimum period: 7.183ns
Maximum Frequency: 139.221MHz
Minimum input arrival time before clock: No path found
Maximum output required time after clock: 2.826ns

2.4. FEEDFORWARD 256 POINT RADIX-2^2 FFT

Device utilization summary:

Selected Device: 5vlx50tff665-2
Slice Logic Utilization:
Number of Slice Registers: 1491 out of 28800  5%
Number of Slice LUTs: 2884 out of 28800  10%
Number used as Logic: 2708 out of 28800  9%

Timing Summary:

Speed Grade: -2
Minimum period: 7.404ns
Maximum Frequency: 135.068MHz
Minimum input arrival time before clock: 1.666ns
Maximum output required time after clock: 2.826ns

3. COMPARISON TABLES

3.1. COMPARISON ON SPEED

TABLE 2. Comparison on speed

<table>
<thead>
<tr>
<th>ARCHITECTURE</th>
<th>FREQUENCY</th>
<th>MIN PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERAL 16 POINT FFT</td>
<td>147.790MHz</td>
<td>6.766ns</td>
</tr>
<tr>
<td>FEEDFORWARD 16 POINT FFT</td>
<td>191.788MHz</td>
<td>5.215ns</td>
</tr>
<tr>
<td>FEEDFORWARD 64 POINT FFT</td>
<td>139.221MHz</td>
<td>7.183ns</td>
</tr>
</tbody>
</table>

3.2. COMPARISON ON AREA

TABLE 3. Comparison on area

<table>
<thead>
<tr>
<th>ARCHITECTURE</th>
<th>NO. OF SLICE REGISTERS</th>
<th>NO. OF SLICE LUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERAL 16 POINT FFT</td>
<td>988  out of 28800 = 3%</td>
<td>1788 out of 28800 = 6%</td>
</tr>
<tr>
<td>FEEDFORWARD 16 POINT FFT</td>
<td>311  out of 28800 = 1%</td>
<td>502 out of 28800 = 1%</td>
</tr>
<tr>
<td>FEEDFORWARD 64 POINT FFT</td>
<td>1113 out of 28800 = 3%</td>
<td>1632 out of 28800 = 5%</td>
</tr>
<tr>
<td>FEEDFORWARD 256 POINT FFT</td>
<td>1491 out of 28800 = 5%</td>
<td>2884 out of 28800 = 10%</td>
</tr>
</tbody>
</table>

4. CONCLUSION

This project extends the use of radix- 2^k to feedforward (MDC) FFT architectures. Indeed, it is shown that feedforward structures are more efficient than feedback ones when several samples in parallel must be processed. In feedforward architectures radix- can be used for any number of parallel samples which is a power of two. In this project we have proposed feedforward architectures for 2 parallel, 4 parallel and 8 parallel samples. Indeed, the number of parallel samples can be chosen arbitrarily depending of the throughput that is required. In this proposed paper we worked on the concept of Fast Fourier Transform (FFT) using both general existing architectures and proposed feedforward architecture and we have implemented in FPGA. Here in this proposed work we mainly concentrate on DIF decomposition. Additionally, both DIF and DIT decompositions can be used. Finally, experimental results show that the designs are efficient both in area and performance than the existing general architectures, being possible to obtain throughputs of the order of G Samples/s as well as very low latencies.
Finally, FFT using feedforward architectures the proposed radix-2^k feedforward architectures not only offer an attractive solution for current applications, but also open up a new research line on feedforward structures in the future.

REFERENCES


